Write a Python program to implement the object-oriented concepts of multiple, Multilevel and Hierarchical Inheritances using your domain applications.

DOMAIN: Netflix Movie Recommendation System.

Multiple Inheritance

Multiple inheritance is said to occur when a class has two or more base (or parent) classes.

```
class User:
    def __init__(self, name):
        self.name = name
        self.preferences = {}

    def set_preference(self, genre, rating):
        self.preferences[genre] = rating

class MovieGenre:
    def __init__(self, genre):
        self.genre = genre
        self.movies = []

    def add_movie(self, movie):
        self.movies.append(movie)

class Movie:
    def __init__(self, title, genre):
        self.title = title
        self.genre = genre

class MovieRecommendationSystem(User, MovieGenre):
    def __init__(self, user, genre):
        User.__init__(self, user)
        MovieGenre.__init__(self, genre)
        self.recommendations = []

    def generate_recommendations(self):
        for movie in self.movies:
            if movie.genre == self.genre and
self.preferences.get(self.genre, 0) >= 3:
                self.recommendations.append(movie)

    def display_recommendations(self):
        print(f"Recommendations for {self.name} based on
{self.genre}:")
        if self.recommendations:
            for movie in self.recommendations:
                print(movie.title)
        else:
            print("No recommendations available.")


user1 = User("Alice")
user1.set_preference("Comedy", 4)
user1.set_preference("Drama", 3)


comedy_genre = MovieGenre("Comedy")
comedy_genre.add_movie(Movie("MovieA", "Comedy"))
comedy_genre.add_movie(Movie("MovieB", "Comedy"))

drama_genre = MovieGenre("Drama")
drama_genre.add_movie(Movie("MovieC", "Drama"))
drama_genre.add_movie(Movie("MovieD", "Drama"))


recommendation_system1 = MovieRecommendationSystem(user1.name,
"Comedy")
recommendation_system1.generate_recommendations()
recommendation_system1.display_recommendations()

recommendation_system2 = MovieRecommendationSystem(user1.name,
"Drama")
recommendation_system2.generate_recommendations()
recommendation_system2.display_recommendations()

Recommendations for Alice based on Comedy:
No recommendations available.
Recommendations for Alice based on Drama:
No recommendations available.

# is creating an error
```

Multi-level Inheritance

Multi-level Inheritance occurs when a class inherits the properties of its parents class, who also inherits properties from its own parent class.

```
class Movie:
    def __init__(self, title, release_year, rating):
        self.title = title
        self.release_year = release_year
        self.rating = rating

    def get_details(self):
        return f"Title: {self.title}, Release Year:
```

```
{self.release_year}, Rating: {self.rating}"


class Genre(Movie):
    def __init__(self, title, release_year, rating, genre):
        super().__init__(title, release_year, rating)
        self.genre = genre

    def get_details(self):
        return f"Title: {self.title}, Release Year:
{self.release_year}, Rating: {self.rating}, Genre: {self.genre}"


class User(Genre):
    def __init__(self, name, age, title, release_year, rating, genre):
        super().__init__(title, release_year, rating, genre)
        self.name = name
        self.age = age

    def get_details(self):
        return f"User: {self.name}, Age: {self.age}, Title:
{self.title}, Release Year: {self.release_year}, Rating:
{self.rating}, Genre: {self.genre}"



movie1 = Movie("Movie A", 2020, 4.5)
genre1 = Genre("Movie B", 2018, 4.0, "Action")
user1 = User("John", 25, "Movie C", 2022, 3.8, "Drama")


print(movie1.get_details())
print(genre1.get_details())
print(user1.get_details())

Title: Movie A, Release Year: 2020, Rating: 4.5
Title: Movie B, Release Year: 2018, Rating: 4.0, Genre: Action
User: John, Age: 25, Title: Movie C, Release Year: 2022, Rating: 3.8,
Genre: Drama
```

Hierarchical Inheritance

When one or more derived classes are created out of a single base class, we can call it Hierarchical Inheritance.

```
class User:
    def __init__(self, username):
        self.username = username
        self.watched_movies = []

    def watch_movie(self, movie):
        self.watched_movies.append(movie)
        print(f"{self.username} watched {movie.title}")

class BasicUser(User):
    def __init__(self, username):
        super().__init__(username)
        self.recommendations = []

    def get_recommendations(self):
        print(f"Fetching recommendations for {self.username}")

        self.recommendations = ["Recommended Movie 1", "Recommended
Movie 2"]
        print(f"Recommended movies for {self.username}: {',
'.join(self.recommendations)}")

class PremiumUser(User):
    def __init__(self, username):
        super().__init__(username)
        self.watchlist = []

    def add_to_watchlist(self, movie):
        self.watchlist.append(movie)
        print(f"{movie.title} added to {self.username}'s watchlist")


class Movie:
    def __init__(self, title):
        self.title = title


user1 = BasicUser("User1")
user2 = PremiumUser("User2")

movie1 = Movie("Movie 1")
movie2 = Movie("Movie 2")

user1.watch_movie(movie1)
user2.watch_movie(movie2)

user1.get_recommendations()
user2.add_to_watchlist(movie1)

print(f"{user1.username}'s watched movies: {', '.join(movie.title for
movie in user1.watched_movies)}")
print(f"{user2.username}'s watchlist: {', '.join(movie.title for movie
in user2.watchlist)}")
```

```
User1 watched Movie 1
User2 watched Movie 2
Fetching recommendations for User1
Recommended movies for User1: Recommended Movie 1, Recommended Movie 2
Movie 1 added to User2's watchlist
User1's watched movies: Movie 1
User2's watchlist: Movie 1
```